NISTIR 86-3473-1

The Internal Revenue Service Post-Of-Duty Location Modeling System: Programmer's Manual for FORTRAN Driver Version 5.0

Paul D. Domich
Richard H. F. Jackson
Marjorie A. McClain

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
(Formerly National Bureau of Standards)
Center for Computing and Applied Mathematics
Gaithersburg, MD 20899

February 1989

A Report to:
The Research Division
Internal Revenue Service
Washington, DC 20224

NISTIR 86-3473-1

The Internal Revenue Service Post-Of-Duty Location Modeling System: Programmer's Manual for FORTRAN Driver Version 5.0

Paul D. Domich
Richard H. F. Jackson
Marjorie A. McClain

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
(Formerly National Bureau of Standards)
Center for Computing and Applied Mathematics
Gaithersburg, MD 20899

February 1989

National Bureau of Standards became the National Institute of Standards and Technology on August 23, 1988, when the Omnibus Trade and Competitiveness Act was signed. NIST retains all NBS functions. Its new programs will encourage improved use of technology by U.S. industry.

A Report to:
The Research Division
Internal Revenue Service
Washington, DC 20224

U.S. DEPARTMENT OF COMMERCE
C. William Verity, Secretary
Ernest Ambler, Acting Under Secretary for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Raymond G. Kammer, Acting Director

ABSTRACT

This report is a programmer's manual for a microcomputer package which was designed by the National Institute of Standards and Technology to assist the Internal Revenue Service in choosing locations for its posts-of-duty which will minimize costs to the IRS and to the taxpayer. The package was written in two sections of code, one in FORTRAN and the other in PASCAL. This manual describes the FORTRAN driver which handles graphics displays and controls input and output for the solution procedure.

TABLE OF CONTENTS

Section I: Introduction

This paper is one of a series of reports documenting the Internal Revenue Service Post-of-Duty Location Modeling System, which was developed for the IRS Research Division by the Center for Computing and Applied Mathematics of the National Institute of Standards and Technology. The reports in the series are as follows.

1) The Internal Revenue Service Post-of-Duty Location Modeling System: Final Report

   This report describes the post-of-duty location problem and its mathematical model. It discusses the types of data which are considered in calculating costs, describes the methods used to solve the location problem, and gives a brief introduction to the computer implementation of the model. (NIST Contact: Paul D. Domich)

2) The Internal Revenue Service Post-of-Duty Location Modeling System: User's Manual

   This report is a user's guide for the post-of-duty location computer system. It gives hardware and software requirements, instructions for installing the system, descriptions of data files, and detailed instructions for operating the system. (NIST Contact: Marjorie A. McClain)

3) The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for FORTRAN Driver

   The post-of-duty location program is written in two sections of code, one in FORTRAN and the other in PASCAL. This report describes the FORTRAN driver which handles graphics displays and controls input and output for the solution procedure. The report includes an alphabetical list of the FORTRAN routines, describing the purpose, the calling sequence and the variables of each routine. (NIST Contact: Marjorie A. McClain)

4) The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for PASCAL Solver

   This report describes the second part of the post-of-duty location program, the PASCAL solver. It discusses the algorithms and data structures used to solve a location problem. (NIST Contact: Paul D. Domich)

It is assumed that the reader of this programmer's manual is thoroughly familiar with IBM-PC DOS, with the FORTRAN language, and with the POD location system user's manual.

Section II: Software Requirements

The following software is required to be able to use and make changes to the POD location system programs.

1) IBM-PC DOS (Version 2.1 or later)
2) IBM Professional FORTRAN (Version 1.19)
3) Turbo PASCAL (Version 5.0)
4) IBM Graphical Kernel System (Version 1.00)
   (GKS includes the Virtual Device Interface and device drivers. See the user's manual for information on how to set up the device drivers in the AUTOEXEC.BAT and CONFIG.SYS files.)
5) Source code for the POD location system, contained in the following files.
   Geographic Data Initialization Programs --
      GDTINST1.FOR
      GDTINST2.FOR
   System Driver Programs --
      LOCATE.BAT
      DRIVER.FOR
      GKSUTIL.FOR
      IOSUBS.FOR
      MAPSUBS.FOR
      MENUSUBS.FOR
   System Solver Programs --
      SOLVER.PAS
      DSTRUCT.PAS
      FIVCLR.PAS
      GREEDY.PAS
      INIT.PAS
      INTCHG.PAS
      LGRN.PAS
      PODCLR.PAS
6) Executable code for the POD location system, contained in the following files.
   Geographic Data Initialization Programs --
      GDTINST1.EXE
      GDTINST2.EXE
   System Driver Programs --
      LOCATE.BAT
      DRIVER.EXE
   System Solver Program --
      SOLVER.EXE

Only a subset of this software is required for a user who just wants to run the system and not make any changes to it. See the user's manual for user software requirements and also for general hardware requirements.

Note: Reference to a tradename or product in this report does not imply endorsement by the National Institute of Standards and Technology.

Section III: Overview of FORTRAN Routines

The POD location system requires the use of geographic data files obtained from Geographic Data Technology, Inc. (GDT). This data must be processed before it is distributed to the end user, in order to convert it to a useable format. The files GDTINST1.FOR and GDTINST2.FOR contain the source code for processing the GDT data. The use of these programs is described more completely in "Instructions for Preparing the IRS Post-of-Duty Location System for Distribution to District Offices".

The POD location system itself is written in two separate sections of code. One section is written in FORTRAN and contains the code for drawing maps and initializing location problems. It allows the user to display workload, set up a location problem, display a map of the solution, and create a report file. The second section is written in PASCAL and contains the code for solving a location problem. Data is passed between the two sections of code using files.

A batch file called LOCATE.BAT controls the flow of execution between the two sections of code. LOCATE.BAT first checks to see if files called ERRORS.GKS and EXITFILE.BAT exist from a previous run. If so, it erases them. Then LOCATE.BAT enters a loop which passes control between the FORTRAN section (stored in DRIVER.EXE) and the PASCAL section (stored in SOLVER.EXE). DRIVER is always entered first. Then, depending on actions taken by the user, either SOLVER is entered or an exit is taken from the loop. In the first case, after SOLVER has completed, control returns to DRIVER and the loop repeats. In the second case, a file called EXITFILE.BAT is created by DRIVER. It contains instructions for deleting work files created during the run and for printing the report file. This new batch file is executed, and then LOCATE.BAT terminates.

The FORTRAN source code is contained in five files: DRIVER.FOR, GKSUTIL.FOR, IOSUBS.FOR, MAPSUBS.FOR and MENUSUBS.FOR. DRIVER.FOR contains the main program DRIVER along with most of the high-level subroutines. The other files contain various lower-level subroutines. A complete description of each subroutine is given in Section IV.

If a change is made in a subroutine, it must be recompiled by typing "PROFORT filename", where "filename" is the name of the source code file containing the changed routine. Then the program must be relinked by typing "LINK DRIVER GKSUTIL IOSUBS MAPSUBS MENUSUBS \GKS\PFGKS,,NUL, \PROFORT\PROFORT \GKS\GKS \GKS\PFGKS /S:5000 /X:350". (This assumes that the POD location system source files are stored in the current directory, Professional FORTRAN files are in a directory called \PROFORT, and GKS files are in a directory called \GKS. These path names may be changed.) The result is a new executable file called DRIVER.EXE.

The PASCAL solver routines are described in "The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for PASCAL Solver".

Section IV: Description of Individual FORTRAN Routines

This section contains a brief description of each of the FORTRAN routines. Information is provided on the purpose of the routine, the name of the file containing its source code, names of input and output variables, and names of programs called by the routine and programs which call it. The routines are listed in alphabetical order.

SUBROUTINE ADJ(IERR,ID,MAXIND,NINDEX,INDEX,MAXZPS,PNTR):

This subroutine reads the GDT zip code boundary file (in DIME format) and creates a binary direct-access adjacency file ADJACENT.id. For each zip code that appears in both the centroid file and the workload file, two records are created in the adjacency file. The first record contains the index of the zip code, the number of adjacent zip codes, and the zip code number. The second record contains a list of the adjacent zip code indices.

Note: This routine should be compiled with the Profort /B option because of the large size of the index array.

Source Code Location --
GDTINST2.FOR

Input Variables --

ID:	CHARACTER*2
	Two-digit district identification number
MAXIND:	INTEGER*4
	Maximum allowed number of elements in INDEX array
MAXZPS:	INTEGER*4
	Maximum allowed number of elements in PNTR array

Output Variables --

IERR:	INTEGER*4
	Error flag --
	IERR=0 for normal return
	IERR=1 if an error was encountered
NINDEX:	INTEGER*4
	Number of elements in index array
INDEX(NINDEX):	INTEGER*4
	Array of zip code index numbers --
	INDEX(I)=J<>0 means that zip code MINZIP+I-1
	appears in both the centroid file and the
	workload file and is assigned index number J
PNTR(MAXZPS):	INTEGER*4
	Array of pointers from index numbers to positions
	of zip codes in centroid file

Programs Called --
COMP

Calling Programs --
GDTINST2

SUBROUTINE BEEP:

This subroutine beeps the speaker.

Source Code Location --
GKSUTIL.FOR

Programs Called --
None

Calling Programs --
STEMAP
ZIPMAP

SUBROUTINE BORDER(XMIN,XMAX,YMIN,YMAX):

This subroutine draws a border around the current screen window. The provided window limits are slightly reduced before the border is drawn; otherwise some sides of the border may not appear because of roundoff error in the conversion from world coordinates to screen coordinates. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
XMIN: REAL*4
Smallest x value of world coordinates
XMAX: REAL*4
Largest x value of world coordinates
YMIN: REAL*4
Smallest y value of world coordinates
YMAX: REAL*4
Largest y value of world coordinates

Programs Called --
BOX
GSPLCI

Calling Programs --
ERSMNU
MAPKEY
MENU20
MENU21
MENU22
MENU23
SPLTWN
TOPMNU
ZIPMAP

SUBROUTINE BOX(XMIN,XMAX,YMIN,YMAX):

This subroutine draws a box. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables
XMIN: REAL*4
      X-coordinate of lower left corner (in world coordinates)
XMAX: REAL*4
      X-coordinate of upper right corner (in world coordinates)
YMIN: REAL*4
      Y-coordinate of lower left corner (in world coordinates)
YMAX: REAL*4
      Y-coordinate of upper right corner (in world coordinates)

Programs Called --
GPL

Calling Programs --
BORDER
MENU20
MENU21
```

SUBROUTINE CENSRT(IERR,ID,MAXM,M,XCENT,YCENT,INDEX,PNT,ZIP5):

This subroutine reads the centroid file CENTROID.id and calls a heapsort program to sort the centroids according to increasing x and y coordinates. (This is done to speed up the process of searching for zip codes on a map. It also speeds up the drawing of maps.) Arrays are set up to store the following information from the file: centroids, pointers to the zip code boundary file, zip code index numbers, and five-digit zip code numbers. This subroutine is executed whenever the program DRIVER is entered.

```
Source Code Location --
DRIVER.FOR

Input Variables --
ID:        CHARACTER*2
           District number
MAXM:      INTEGER*4
           Maximum number of zip codes allowed

Output Variables --
IERR:      INTEGER*4
           Error flag --
           IERR=0 for normal return
           IERR=1 if an error was encountered
M:         INTEGER*4
           Number of zip codes
XCENT(M):  REAL*4
           Array containing x-coordinates of centroids,
           sorted according to increasing x and y
YCENT(M):  REAL*4
           Array containing y-coordinates of centroids,
           sorted according to increasing x and y
INDEX(M):  INTEGER*2
```

```
                Zip code index array --
                INDEX(I) is the original index of the zip code with
                centroid (XCENT(I),YCENT(I)) before sorting
        PNT(M):  INTEGER*2
                Pointer array --
                PNT(I) points to the beginning of information on the
                zip code with centroid (XCENT(I),YCENT(I)) in the zip
                code boundary file
        ZIP5(M): INTEGER*4
                Array of five-digit zip codes, in original sequential order

        Programs Called --
        PAUSE
        SORT

        Calling Programs --
        DRIVER
```

SUBROUTINE CENTRD(IERR,ID,NZIPS,ZIP5,PNTR,NCENT):

This subroutine reads the GDT zip code inventory file and creates a binary direct-access centroid file CENTROID.id. Zip codes with zero area (point zips) are not included in the centroid file. A pointer is included with each centroid which points to the corresponding zip code in the zip code boundary file ZIPCODE.id. A post office name file PONAMES.id is also created, listing each zip code and its name.

```
        Source Code Location --
        GDTINST1.FOR

        Input Variables --
        ID:           CHARACTER*2
                      Two-digit district identification number
        NZIPS:        INTEGER*4
                      Number of distinct zip codes in the zip code
                      boundary file
        ZIP5(NZIPS): INTEGER*4
                      Array of five-digit zip code numbers, in sequential
                      order
        PNTR(NZIPS): INTEGER*4
                      Pointer into boundary file --
                      PNTR(I) is the record number of the beginning of
                      information on zip code ZIP5(I) in the boundary file

        Output Variable --
        IERR:         INTEGER*4
                      Error flag --
                      IERR=0 for normal return
                      IERR=1 if an error was encountered
        NCENT:        INTEGER*4
                      Number of distinct zip codes which appear in both the
                      inventory file and the boundary file

        Programs Called --
```

None

Calling Programs --
GDTINST1

SUBROUTINE CHOICE(WKID,CHDNR):

This subroutine initializes choice mode (function keys). To use the function keys, call GRQCH ("Request Choice"). (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID: INTEGER*2
Workstation identifier

Output Variable --
CHDNR: INTEGER*2
Choice device number

Programs Called --
GSCHM

Calling Programs --
DISPLY
DRIVER

SUBROUTINE CLSGKS(WKID):

This subroutine closes GKS and resets the screen mode for text. (This requires the presence of ANSI.SYS in the CONFIG.SYS file. See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID: INTEGER*2
Workstation identifier

Programs Called --
GCLKS
GCLWK
GDAWK

Calling Programs --
DISPLY
DRIVER
MAPKEY
STEMAP
STYLE
ZIPMAP

SUBROUTINE COMP(IERR,ID,MINZIP,MAXIND,NINDEX,INDEX, MAXZPS,PNTR):

This subroutine compares zip codes in the centroid file with zip codes in the workload file. It assigns an index number to each zip code which appears in both files. It also creates a file COMPARE.id listing zip codes which appear in only one of the files.

Source Code Location --
GDTINST2.FOR

Input Variables --

ID:	CHARACTER*2
	Two-digit district identification number
MAXIND:	INTEGER*4
	Maximum allowed number of elements in INDEX array
MAXZPS:	INTEGER*4
	Maximum allowed number of elements in PNTR array

Output Variables --

IERR:	INTEGER*4
	Error flag --
	IERR=0 for normal return
	IERR=1 if an error was encountered
MINZIP:	INTEGER*4
	Minimum zip code in centroid file
NINDEX:	INTEGER*4
	Number of elements in INDEX array
INDEX(NINDEX):	INTEGER*4
	Array of zip code index numbers --
	INDEX(I)=J<>0 means that zip code MINZIP+I-1
	appears in both the centroid file and the
	workload file and is assigned index number J
PNTR(MAXZPS):	INTEGER*4
	Array of pointers from index numbers to positions
	of zip codes in centroid file

Programs Called --
None

Calling Programs --
ADJ

SUBROUTINE CONVRT(INPUT,OUTPUT):

This subroutine converts a one- or two-digit integer into a two-character string (used for printing out the date and time).

Source Code Location --
IOSUBS.FOR

Input Variable --

INPUT:	INTEGER*2
	Integer between 0 and 99

Output Variable --
OUTPUT: CHARACTER*2
        Two-character string version of input integer

Programs Called --
None

Calling Programs --
REPFIL

FUNCTION COSTFN(NCLASS,SWITCH,WRK,IRSWT,IRSFCT,TXPWT,TXPFCT, MILCST,TRVDIF,DIST,SQFCST,SQFT):

This function calculates the cost of assigning a zip code to a POD site. The cost is made up of travel costs and office space costs. Travel costs are based on the Individual Master File (IMF) and Business Master File (BMF) workload generated by the zip code, its distance from the POD site, and the cost per mile. However, the cost may be weighted by several user-supplied factors. Office space costs are based on the amount of space required to handle the workload and the rental cost per square foot for office space at the POD site. For further information on how costs are computed, see "The Internal Revenue Service Post-of-Duty Location Modeling System: Final Report".

Source Code Location --
DRIVER.FOR

Input Variables --

NCLASS:	INTEGER*4 Number of workload columns in workload file (not including display columns)
SWITCH(NCLASS):	INTEGER*4 Array indicating which columns of the workload file are to be included in the cost calculation
WRK(NCLASS):	INTEGER*4 Workload array for current zip code
IRSWT:	REAL*4 Weight (nonnegative real number) to be assigned to IRS travel costs
IRSFCT(NCLASS):	REAL*4 Array of IRS trip factors
TXPWT:	REAL*4 Weight (nonnegative real number) to be assigned to taxpayer travel costs
TXPFCT(NCLASS):	REAL*4 Array of taxpayer trip factors
MILCST:	REAL*4 Mileage cost ($/mile)
TRVDIF:	REAL*4 Travel difficulty factor for trips between current zip code and current POD site
DIST:	REAL*4 Distance between current zip code and current POD site (miles)

```
SQFCST:          REAL*4
                 Office space cost for current POD site
                 ($/square foot/year)
SQFT:            REAL*4
                 Office space factor to be multiplied by total
                 TDI's and TDA's (square feet)

Output Variable --
COSTFN:          REAL*8
                 Cost of assigning current zip code to current POD
                 site ($)

Programs Called --
None

Calling Programs --
SOLVE
```

SUBROUTINE CRSBOX(WKID,TRN,STAT,XMIN,XMAX,YMIN,YMAX):

This subroutine creates a cursor box. To use the cursor, first move the crosshair cursor to one corner of the desired box and enter this point. Then a rectangular cursor will appear which can be used to enter the opposite corner of the box. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
      Workstation identifier
TRN:  INTEGER*2
      Transformation number

Output Variables --
STAT: INTEGER*2
      Cursor status indicator
      STAT=0: Error reading cursor locations
      STAT=1: No errors encountered
XMIN: REAL*4
      X coordinate of lower left corner of box
XMAX: REAL*4
      X coordinate of upper right corner of box
YMIN: REAL*4
      Y coordinate of lower left corner of box
YMAX: REAL*4
      Y coordinate of upper right corner of box

Programs Called --
GINLC
GPREC
GQNT
GRQLC
```

Calling Programs --
ZOOMIN

SUBROUTINE CURSOR(WKID,TRN,IPX,IPY):

This subroutine initializes a cursor. To use the cursor, call GRQLC ("Request Locator"). (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
Workstation identifier
TRN: INTEGER*2
Transformation number
IPX: REAL*4
Initial x position of cursor in world coordinates
IPY: REAL*4
Initial y position of cursor in world coordinates

Programs Called --
GINLC
GPREC

Calling Programs --
MAPKEY

SUBROUTINE DISPLY(IERR,ID,PALETT,NCLRS,MENU,COLOR,MODIFY,
M,XCENT,YCENT,INDEX,PNT,ZIPCLR,ZIP5,
MAXPLT,XPLOT,YPLOT,LONG,PERM):

This subroutine is the driver for the map-drawing routines. It uses GKS to display a state map and then allows the user to zoom in or modify the colors on the map.

Source Code Location --
MAPSUBS.FOR

Input Variables --
ID: CHARACTER*2
District number
PALETT: INTEGER*2
Palette identifier (1 or 2)
NCLRS: INTEGER*2
Number of colors in menu
(only used if MODIFY=.TRUE.)
MENU: INTEGER*2
Menu type switch (for coloring) --
MENU=0 for general coloring menu
MENU=1 for menu of POD types
MENU=2 for title of solution map
MENU=3 for key to workload map

```
COLOR:          LOGICAL
                Color indicator --
                COLOR=.TRUE. if an array of zip code colors is
                     provided
                COLOR=.FALSE. otherwise
MODIFY:         LOGICAL
                Color modification indicator --
                MODIFY=.TRUE. if the array of zip code colors is
                      allowed to be modified
                MODIFY=.FALSE. otherwise
M:              INTEGER*4
                Number of zip codes
XCENT(M):       REAL*4
                Array containing x-coordinates of centroids,
                sorted according to increasing x and y
YCENT(M):       REAL*4
                Array containing y-coordinates of centroids,
                sorted according to increasing x and y
INDEX(M):       INTEGER*2
                Zip code index array --
                INDEX(I) is the original index of the zip code
                with centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):         INTEGER*2
                Pointer array --
                PNT(I) points to the beginning of information on
                the zip code with centroid (XCENT(I),YCENT(I)) in
                the zip code boundary file
ZIPCLR(M):      INTEGER*2
                Zip code color array --
                ABS(ZIPCLR(INDEX(I))) is the color (a number from
                1 to 9) of the zip code with centroid (XCENT(I),
                YCENT(I))
                (only input if COLOR=.TRUE.)
                Positive value: Centroid drawn as asterisk
                Negative value: Centroid drawn as small box
ZIP5(M):        INTEGER*4
                Array of five-digit zip codes, in original
                sequential order
MAXPLT:         INTEGER*4
                Maximum number of plotting points allowed in a
                map region

Output Variables --
IERR:           INTEGER*4
                Error flag --
                IERR=0 for normal return
                IERR=1 if an error was encountered
COLOR:          LOGICAL
                Set to .TRUE. if color array was created
ZIPCLR(M):      INTEGER*2
                Modified zip code color array
                (same as input if MODIFY=.FALSE.)
XPLOT(2*MAXPLT): REAL*4
                Work array containing x coordinates of points to
```

```
                    plot in a map region
YPLOT(2*MAXPLT):    REAL*4
                    Work array containing y coordinates of points to
                    plot in a map region
LONG(2*MAXPLT):     REAL*4
                    Work array
PERM(2*MAXPLT):     INTEGER*2
                    Work array

Programs Called --
CHOICE
CLSGKS
GCLRWK
GRQCH
GSELNT
GSTXCI
GTX
MAPKEY
MARKZP
MENU11
MENU12
OPNGKS
STEMAP
ZIPMAP
ZOOMIN

Calling Programs --
PODMAP
SOLMAP
WRKMAP
```

PROGRAM DRIVER:

This program is the driver for the IRS post-of-duty location system. It first writes a title page on the screen, initializes the report file, and sets up data structures for storing zip code information. Then it uses GKS to write the top-level menu on the screen and allows the user to choose which phase of a location problem to work on. The possible choices are:

(1) Exit.
(2) Display workload.
(3) Display or modify initial POD sites.
(4) Solve for optimal POD locations.
(5) Display optimal POD locations.
(6) Report zip code and staff assignments.

The driver terminates in one of two possible ways:

(1) The user chooses "F1 - Exit" from the top menu. In this case, a file called EXITFILE.BAT is created which signals the batch program LOCATE.BAT to leave the POD location system.
(2) The user chooses "F4 - Solve for optimal POD locations" from the top menu. In this case, LOCATE.BAT passes control to the solution procedure SOLVER.EXE. When the solution program is finished, control returns to DRIVER.EXE.

The following data files must exist to run the package (the "id" extension should be replaced by the district identification number):

(1) OUTLINE.id: State outline file
(2) ZIPCODE.id: Zip code boundary file
(3) CENTROID.id: Zip code centroid file
(4) PODSITE.id: POD site and fixed cost file
(5) WORKLOAD.id: Workload file

(See the user's manual for information on the format of these files.)

The following common block is required by the graphics package GKS:

```
COMMON /GRACOM/ SIZE,INTARY
```

SIZE is an INTEGER*4 variable set equal to 9000, and INTARY is an INTEGER*4 array of length 9000.

Source Code Location --
DRIVER.FOR

Programs Called --
CENSRT
CHOICE
CLSGKS
EXIT
GRQCH
HEADER
OPNGKS
PODMAP
REPFIL
REPSOL
SOLMAP
SOLVE
TOPMNU
WRKMAP

SUBROUTINE ERSMNU:

This subroutine erases a menu from the left window of the screen.

Source Code Location --
MENUSUBS.FOR

Programs Called --
BORDER
GFA
GQCNTN
GSELNT
STYLE

Calling Programs --
MAPKEY
ZOOMIN

SUBROUTINE EXIT(ID):

This subroutine is executed when the user chooses "F1 - Exit" from the top menu. It creates a batch file called EXITFILE.BAT which contains instructions for deleting work files created by the rest of the program. It also saves modifications to the PODSITE.id file and sends the report file to the line printer if the user requests it. After this subroutine has been completed, control returns to the batch file LOCATE.BAT, which checks to see if EXITFILE.BAT exists, executes it, and leaves the system.

Source Code Location --
DRIVER.FOR

Input Variable --
ID: CHARACTER*2
District number

Programs Called --
None

Calling Programs --
DRIVER

SUBROUTINE GACWK(WKID):

GKS routine -- "ACTIVATE WORKSTATION"
(See GKS manual for further information.)

Calling Programs --
OPNGKS

FUNCTION GCDIST(LAT1,LONG1,LAT2,LONG2):

This function calculates the great circle distance (in miles) between two points on the earth, given the latitude and longitude of each point. The formula was provided by Rand McNally.

Source Code Location --
DRIVER.FOR

Input Variables --
LAT1: REAL*4
Latitude of first point
LONG1: REAL*4
Longitude of first point
LAT2: REAL*4
Latitude of second point
LONG2: REAL*4
Longitude of second point

Output Variable --
GCDIST: REAL*4
Distance (in miles) between first and second points

Programs Called --
None

Calling Programs --
SOLVE
SPLTWN

SUBROUTINE GCLKS:

GKS routine -- "CLOSE KERNEL SYSTEM"
(See GKS manual for further information.)

Calling Programs --
CLSGKS

SUBROUTINE GCLRWK(WKID,COFL):

GKS routine -- "CLEAR WORKSTATION"
(See GKS manual for further information.)

Calling Programs --
DISPLY
SPLTWN
ZOOMIN

SUBROUTINE GCLWK(WKID):

GKS routine -- "CLOSE WORKSTATION"
(See GKS manual for further information.)

Calling Programs --
CLSGKS

SUBROUTINE GDAWK(WKID):

GKS routine -- "DEACTIVATE WORKSTATION"
(See GKS manual for further information.)

Calling Programs --
CLSGKS

PROGRAM GDTINST1:

This program converts Geographic Data Technology (GDT) map data into a format to be used by the IRS POD location system. Following the use of this program, GDTINST2 should be run to complete the installation. This program should be used only by the person preparing the data files for distribution, not by the end user of the system.

Source Code Location --
GDTINST1.FOR

Programs Called --
CENTRD
OUTLIN
ZIPBND

PROGRAM GDTINST2:

This program creates a zip code adjacency file using GDT's DIME format boundary file. It also compares zip codes in the IRS workload file with zip codes in the centroid file (created by GDTINST1), and creates a file listing any mismatches. This program should be used only by the person preparing the data files for distribution, not by the end user of the system.

Note: This program should be compiled with the Profort /B option.

Source Code Location --
GDTINST2.FOR

Programs Called --
ADJ

SUBROUTINE GEOFAC(DIFZIP,DIFPOD,DIFFAC,K1,NREC,ID):

This subroutine allows the user to enter factors indicating the degree of difficulty of travel between a zip code and a POD site. The values are stored in the file DIFFACT.id.

Source Code Location --
IOSUBS.FOR

Input Variables --
K1: INTEGER*4
Maximum number of non-unit travel difficulty factors
ID: CHARACTER*2
District number

Output Variables --
DIFZIP(NREC): INTEGER*4
five-digit zip code array for use with travel difficulty factors
DIFPOD(NREC): INTEGER*4
POD zip code array for use with travel difficulty factors
DIFFAC(NREC): REAL*4
Array of travel difficulty factors --
DIFFAC(I) is the factor associated with zip code DIFZIP(I) and POD site DIFPOD(I)
NREC: INTEGER*4
Number of non-unit travel difficulty factors

Programs Called --
READBF

Calling Programs --
SOLVE

SUBROUTINE GETDAT(IYEAR,IMONTH,IDAY):

Professional FORTRAN routine -- "GET DATE"

(See Professional FORTRAN manual for further information.)

Calling Programs --
REPFIL

SUBROUTINE GETTIM(IHOUR,IMINUTE,ISECOND,IHUNDREDSECOND):

Professional FORTRAN routine -- "GET TIME"
(See Professional FORTRAN manual for further information.)

Calling Programs --
REPFIL

SUBROUTINE GFA(N,PX,PY):

GKS routine -- "FILL AREA"
(See GKS manual for further information.)

Calling Programs --
ERSMNU
MAPKEY
MENU20
MENU21
MENU22
MENU23
ZIPMAP

SUBROUTINE GINKS(NTYPES,WNAMES,WTYPES,VERNUM):

GKS routine -- "INITIALIZE GKS"
(See GKS manual for further information.)

Calling Programs --
OPNGKS

**SUBROUTINE GINLC(WKID,LCDNR,TNR,IPX,IPY,PET,XMIN,XMAX,YMIN,YMAX,
LDR,DATREC):**

GKS routine -- "INITIALIZE LOCATOR"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
CURSOR

**SUBROUTINE GINST(WKID,STDNR,ISTR,PET,XMIN,XMAX,YMIN,YMAX,BUFLEN,
INIPOS,LDR,DATREC):**

GKS routine -- "INITIALIZE STRING"
(See GKS manual for further information.)

Calling Programs --
MARKZP
ZOOMIN

SUBROUTINE GOPKS(ERRFIL,SIZE):

GKS routine -- "OPEN KERNEL SYSTEM"
(See GKS manual for further information.)

Calling Programs --
OPNGKS

SUBROUTINE GOPWK(WKID,CONID,WTYPE):

GKS routine -- "OPEN WORKSTATION"
(See GKS manual for further information.)

Calling Programs --
OPNGKS

SUBROUTINE GPL(N,PX,PY):

GKS routine -- "POLYLINE"
(See GKS manual for further information.)

Calling Programs --
BOX
MAPKEY
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
STEMAP
TOPMNU
ZIPMAP

SUBROUTINE GPM(N,PX,PY):

GKS routine -- "POLYMARKER"
(See GKS manual for further information.)

Calling Programs --
MAPKEY
MARKZP
STEMAP
ZIPMAP

SUBROUTINE GPREC(IL,IA,RL,RA,SL,MSTR,STR,MLDR,ERRIN,LDR,DATREC):

GKS routine -- "PACK DATA RECORD"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
CURSOR

SUBROUTINE GQCF(WTYPE,ERRIND,NCOLI,COLA,NPCI):

GKS routine -- "INQUIRE COLOR FACILITIES"
(See GKS manual for further information.)

Calling Programs --
MARKZP
SETCOL
STEMAP
STYLE
TOPMNU
ZIPMAP
ZOOMIN

SUBROUTINE GQCNTN(ERRIND,CTNR):

GKS routine -- "INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER"
(See GKS manual for further information.)

Calling Programs --
ERSMNU
MARKZP
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23

SUBROUTINE GQMDS(WTYPE,ERRIND,DCUNIT,RX,RY,LX,LY):

GKS routine -- "INQUIRE MAXIMUM DISPLAY SURFACE SIZE"
(See GKS manual for further information.)

Calling Programs --
MARKZP
SPLTWN
WINDOW
ZOOMIN

SUBROUTINE GQNT(TNR,ERRIND,WINDOW,VIEWPT):

GKS routine -- "INQUIRE NORMALIZATION TRANSFORMATION"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
MAPKEY
MARKZP

SUBROUTINE GQTXX(WKID,PX,PY,STR,ERRIND,CPX,CPY,TXEXPX,TXEXPY):

GKS routine -- "INQUIRE TEXT EXTENT"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
ZIPMAP

SUBROUTINE GRQCH(WKID,CHDNR,STAT,CHNR):

GKS routine -- "REQUEST CHOICE"
(See GKS manual for further information.)

Calling Programs --
DISPLY
DRIVER
MAPKEY

SUBROUTINE GRQLC(WKID,LCDNR,STAT,TNR,PX,PY):

GKS routine -- "REQUEST LOCATOR"
(See GKS manual for further information.)

Calling Programs --
CRSBOX
MAPKEY

SUBROUTINE GRQST(WKID,STDNR,STAT,LOSTR,STR):

GKS routine -- "REQUEST STRING"
(See GKS manual for further information.)

Calling Programs --
MARKZP
ZOOMIN

SUBROUTINE GSCHM(WKID,CHDNR,MODE,ESW):

GKS routine -- "SET CHOICE MODE"
(See GKS manual for further information.)

Calling Programs --
CHOICE

SUBROUTINE GSCR(WKID,CI,CR,CG,CB):

GKS routine -- "SET COLOR REPRESENTATION"
(See GKS manual for further information.)

Calling Programs --
SETCOL

SUBROUTINE GSELNT(TNR):

GKS routine -- "SELECT NORMALIZATION TRANSFORMATION"
(See GKS manual for further information.)

Calling Programs --

DISPLY
ERSMNU
MAPKEY
MARKZP
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
SPLTWN
WINDOW
ZOOMIN

SUBROUTINE GSFACI(COLI):

GKS routine -- "SET FILL AREA COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
STYLE

SUBROUTINE GSFAIS(INTS):

GKS routine -- "SET FILL AREA INTERIOR STYLE"
(See GKS manual for further information.)

Calling Programs --
STYLE

SUBROUTINE GSFASI(STYLI):

GKS routine -- "SET FILL AREA STYLE INDEX"
(See GKS manual for further information.)

Calling Programs --
STYLE

SUBROUTINE GSMK(MTYPE):

GKS routine -- "SET POLYMARKER TYPE"
(See GKS manual for further information.)

Calling Programs --
MAPKEY
MARKZP
STEMAP
ZIPMAP

SUBROUTINE GSPLCI(COLI):

GKS routine -- "SET POLYLINE COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
BORDER
MAPKEY
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
STEMAP
TOPMNU
ZIPMAP

SUBROUTINE GSPMCI(COLI):

GKS routine -- "SET POLYMARKER COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
MAPKEY
MARKZP
MATCH
STEMAP
ZIPMAP

SUBROUTINE GSSTM(WKID,STDNR,MODE,ESW):

GKS routine -- "SET STRING MODE"
(See GKS manual for further information.)

Calling Programs --
MARKZP

SUBROUTINE GSTXCI(COLI):

GKS routine -- "SET TEXT COLOR INDEX"
(See GKS manual for further information.)

Calling Programs --
DISPLY
MAPKEY
MARKZP
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
TOPMNU
ZIPMAP
ZOOMIN

SUBROUTINE GSVP(TNR,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET VIEWPORT"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GSWKVP(WKID,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET WORKSTATION VIEWPORT"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GSWKWN(WKID,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET WORKSTATION WINDOW"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GSWN(TNR,XMIN,XMAX,YMIN,YMAX):

GKS routine -- "SET WINDOW"
(See GKS manual for further information.)

Calling Programs --
SPLTWN
WINDOW

SUBROUTINE GTX(PX,PY,CHARS):

GKS routine -- "TEXT"
(See GKS manual for further information.)

Calling Programs --
DISPLY
MAPKEY
MARKZP
MENU11
MENU12
MENU20
MENU21
MENU22
MENU23
TOPMNU
ZIPMAP
ZOOMIN

SUBROUTINE HEADER:

This subroutine prints a title page on the screen when the user enters the system.

Source Code Location --
IOSUBS.FOR

Programs Called --
None

Calling Programs --
DRIVER

SUBROUTINE MAPKEY(IERR,MODIFY,TRN,PALETT,NCLRS,ID,MENU,MAXPLT, M,XCENT,YCENT,INDEX,PNT,ZIPCLR,XPLOT,YPLOT, LONG,PERM):

This subroutine displays a color key for the current zip code map. It allows the user to locate a zip code on the map with the cursor and have its five-digit number printed on the screen. (To do this, a search of the centroid array is done to find the closest centroid to the cursor position.) If the current map is for modifying POD sites, the routine also allows the colors of zip codes to be modified.

Source Code Location --
MAPSUBS.FOR

Input Variables --

MODIFY:	LOGICAL
	Color modification indicator --
	MODIFY=.TRUE. if the array of zip code colors is allowed to be modified
	MODIFY=.FALSE. otherwise
TRN:	INTEGER*2
	Transformation number
PALETT:	INTEGER*2
	Palette identifier (1 or 2)
NCLRS:	INTEGER*2
	Number of colors in menu
	(only used if MODIFY=.TRUE.)
ID:	CHARACTER*2
	District number
MENU:	INTEGER*2
	Menu type switch --
	MENU=0 for general coloring menu
	MENU=1 for menu of POD types
	MENU=2 for title of solution map
	MENU=3 for key to workload map
M:	INTEGER*4
	Number of zip codes
XCENT(M):	REAL*4
	Array containing x-coordinates of centroids, sorted according to increasing x and y
YCENT(M):	REAL*4
	Array containing y-coordinates of centroids,

```
                    sorted according to increasing x and y
INDEX(M):           INTEGER*2
                    Zip code index array, sorted corresponding to
                    XCENT and YCENT arrays
PNT(M):             INTEGER*2
                    Pointer array, sorted corresponding to XCENT and
                    YCENT arrays
ZIPCLR(M):          INTEGER*2
                    Zip code color array --
                    Positive value: Centroid drawn as asterisk
                    Negative value: Centroid drawn as small box
MAXPLT:             INTEGER*4
                    Maximum number of plotting points allowed in a
                    map region

Output Variables --
IERR:               INTEGER*4
                    Error flag --
                    IERR=0 for normal return
                    IERR=1 if an error was encountered
ZIPCLR(M):          INTEGER*2
                    Zip code color array
XPLOT(2*MAXPLT):    REAL*4
                    Work array containing x coordinates of points to
                    plot in a map region
YPLOT(2*MAXPLT):    REAL*4
                    Work array containing y coordinates of points to
                    plot in a map region
LONG(2*MAXPLT):     REAL*4
                    Work array
PERM(2*MAXPLT):     INTEGER*2
                    Work array

Programs Called --
BORDER
CLSGKS
CURSOR
ERSMNU
GFA
GPL
GPM
GQNT
GRQCH
GRQLC
GSELNT
GSMK
GSPLCI
GSPMCI
GSTXCI
GTX
MENU20
MENU21
MENU22
MENU23
```

REVERS
SEARCH
STYLE
THIN

Calling Programs --
DISPLY

SUBROUTINE MARKZP(ID,M,ZIP5):

This subroutine allows the user to type in a zip code number and have its location marked on the state outline map.

Source Code Location --
MAPSUBS.FOR

Input Variables --
ID: CHARACTER*2
District number
M: INTEGER*4
Number of zip codes
ZIP5(M): INTEGER*4
Array of five-digit zip codes, in original sequential order

Programs Called --
GINST
GPM
GQCF
GQCNTN
GQMDS
GQNT
GRQST
GSELNT
GSMK
GSPMCI
GSSTM
GSTXCI
GTX
REVERS

Calling Programs --
DISPLY

SUBROUTINE MATCH(PALETT,INDEX):

This subroutine sets a polymarker color index which will match the corresponding fill area color index (for an EGA display only). (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
PALETT: INTEGER*2

Palette identifier
INDEX: INTEGER*4
Index (between 1 and 16) of fill area color

Programs Called --
GSPMCI

Calling Programs --
STEMAP

SUBROUTINE MENU11:

This subroutine writes a menu of display options on the left side of the screen. It lists the meaning of the function keys after a state outline map has been drawn.

Source Code Location --
MENUSUBS.FOR

Programs Called --
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX

Calling Programs --
DISPLY

SUBROUTINE MENU12:

This subroutine writes a menu of display options on the left side of the screen. It lists the meaning of the function keys after a zip code map has been drawn.

Source Code Location --
MENUSUBS.FOR

Programs Called --
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX

Calling Programs --
DISPLY

SUBROUTINE MENU20:

This subroutine writes a color key on the left side of the screen. It lists the colors associated with the function keys.

Source Code Location --
MENUSUBS.FOR

Programs Called --
BORDER
BOX
GFA
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Programs --
MAPKEY

SUBROUTINE MENU21:

This subroutine writes a color key on the left side of the screen. It lists the colors and function keys associated with different types of POD sites.

Source Code Location --
MENUSUBS.FOR

Programs Called --
BORDER
BOX
GFA
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Programs --
MAPKEY

SUBROUTINE MENU22:

This subroutine displays a title for the solution map on the left side of the screen.

Source Code Location --
MENUSUBS.FOR

Programs Called --
BORDER
GFA
GPL
GQCNTN

GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Programs --
MAPKEY

SUBROUTINE MENU23:

This subroutine writes a color key on the left side of the screen. It lists the colors and function keys used for displaying workload.

Source Code Location --
MENUSUBS.FOR

Programs Called --
BORDER
GFA
GPL
GQCNTN
GSELNT
GSPLCI
GSTXCI
GTX
STYLE

Calling Programs --
MAPKEY

SUBROUTINE OPNGKS(WKID):

This subroutine opens GKS with one workstation of type "DISPLAY". Error messages will be written to a file called "ERRORS.GKS". (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID: INTEGER*2
Workstation identifier

Programs Called --
GACWK
GINKS
GOPKS
GOPWK

Calling Programs --
DISPLY
DRIVER

SUBROUTINE OPTION(IO,DSTLIM,TXPWGT,IRSWGT,IMF,BMF,SWITCH,NCLASS,
SQFT,MILCST,STAFF1,STAFF2,STAFF3,PODCHG):

This subroutine displays the default options for the cost calculation and allows the user to make changes.

Source Code Location --
IOSUBS.FOR

Input Variables --

IO:	INTEGER*4 Unit number defining where display will appear
DSTLIM:	REAL*4 Default maximum travel distance allowed from zip code to POD
TXPWGT:	REAL*4 Default weight (nonnegative real number) to be assigned to taxpayer travel costs
IRSWGT:	REAL*4 Default weight (nonnegative real number) to be assigned to IRS travel costs
IMF(2):	INTEGER*4 Default array indicating which IRS functions are to be included in the cost calculation for IMF data -- IMF(1): Examination IMF(2): Collection
BMF(2):	INTEGER*4 Default array indicating which IRS functions are to be included in the cost calculation for BMF data -- BMF(1): Examination BMF(2): Collection
NCLASS:	INTEGER*4 Number of workload columns in workload file (not including display columns)
SQFT:	REAL*4 Default office space required per TDI-TDA (square feet)
MILCST:	REAL*4 Default Mileage cost ($/mile)
STAFF1:	INTEGER*4 Default total examination staff available
STAFF2:	INTEGER*4 Default total collection staff available
STAFF3:	INTEGER*4 Default total other staff available

Output Variables

DSTLIM:	REAL*4 Maximum travel distance allowed from zip code to POD
TXPWGT:	REAL*4 Weight (nonnegative real number) to be assigned to

taxpayer travel costs

IRSWGT: REAL*4
Weight (nonnegative real number) to be assigned to IRS travel costs

IMF(2): INTEGER*4
Array indicating which IRS functions are to be included in the cost calculation for IMF data

BMF(2): INTEGER*4
Array indicating which IRS functions are to be included in the cost calculation for BMF data

SWITCH(NCLASS): INTEGER*2
Array indicating which columns of the workload file are to be included in the cost calculation

SQFT: REAL*4
Office space required per TDI-TDA (square feet)

MILCST: REAL*4
Mileage cost ($/mile)

STAFF1: INTEGER*4
Total examination staff available

STAFF2: INTEGER*4
Total collection staff available

STAFF3: INTEGER*4
Total other staff available

PODCHG: LOGICAL
Flag indicating whether POD information has been changed

Programs Called --
READBF
ROUND

Calling Programs --
REPPRB
SOLVE

SUBROUTINE ORDER(XINIT,YINIT,N,X,Y):

This subroutine reorders the interior points of a zip code with a hole in it so that the connector segment joins the first exterior point with the closest interior point. This is done to avoid problems which arise when the GKS shading ("Fill Area") routine is used with a connector segment that crosses the interior hole. There may still be problems with zip codes which contain several holes.

Source Code Location --
GDTINST1.FOR

Input Variables --
XINIT: REAL*4
X coordinate of initial exterior connecting point

YINIT: REAL*4
Y coordinate of initial exterior connecting point

N: INTEGER*4
Number of interior points (boundary of hole)

```
X(N):  REAL*4
       Array containing x coordinates of interior points
Y(N):  REAL*4
       Array containing y coordinates of interior points

Output Variables --
X(N):  REAL*4
       Reordered array of interior x coordinates
Y(N):  REAL*4
       Reordered array of interior y coordinates

Programs Called --
None

Calling Programs --
ZIPBND
```

SUBROUTINE OUTLIN(IERR,ID,XMIN,YMIN,XMAX,YMAX,MAXPTS,X,Y):

This subroutine converts the GDT state boundary file to binary form, thinning out the data in the process.

```
Source Code Location --
GDTINST1.FOR

Input Variables --
ID:         CHARACTER*2
            Two-digit district identification number
MAXPTS:     INTEGER*4
            Maximum number of points allowed in a map region

Output Variables --
IERR:       INTEGER*4
            Error flag --
            IERR=0 for normal return
            IERR=1 if an error was encountered
XMIN:       REAL*4
            Minimum x coordinate of state map
YMIN:       REAL*4
            Minimum y coordinate of state map
XMAX:       REAL*4
            Maximum x coordinate of state map
YMAX:       REAL*4
            Maximum y coordinate of state map
X(MAXPTS):  REAL*4
            Work array containing x coordinates of a map region
Y(MAXPTS):  REAL*4
            Work array containing y coordinates of a map region

Programs Called --
None

Calling Programs --
GDTINST1
```

SUBROUTINE PAUSE:

This subroutine requests the user to press "Enter" to continue. It is used following error messages to allow the user time to see the message.

Source Code Location --
IOSUBS.FOR

Programs Called --
None

Calling Programs --
CENSRT
PODMAP
REPSOL
SOLMAP
SOLVE
WRKMAP

SUBROUTINE PODMAP(IERR,ID,PODSET,M,XCENT,YCENT,INDEX,PNT, ZIP5,ZIPCLR,MAXPLT,XPLOT,YPLOT,LONG,PERM):

This subroutine initializes the display of a map of current and potential POD sites. It is executed when the user chooses "F3 - Display or modify initial POD sites" from the top menu. First it reads the file PODSITE.id (or NEWPOD.id) to find which zip codes are allowed to be POD sites and which zip codes currently are POD sites. Color #1 (background) is assigned to zip codes which cannot be POD sites, color #2 is assigned to potential POD sites, and color #3 is assigned to current POD sites. (Note: A negative color number indicates that the centroid will be drawn as a small box; otherwise the centroid is an asterisk.) Then these colors are passed in an array to the subroutine DISPLY, which draws the map and lets the user change the colors. The new colors are returned in the original array. Color #4 in the new array indicates a zip code which the user has made a fixed POD site.

Source Code Location --
DRIVER.FOR

Input Variables --

ID:	CHARACTER*2 District number
M:	INTEGER*4 Number of zip codes
PODSET:	LOGICAL Flag indicating whether or not POD colors have been previously set
XCENT(M):	REAL*4 Array containing x-coordinates of centroids, sorted according to increasing x and y
YCENT(M):	REAL*4 Array containing y-coordinates of centroids, sorted according to increasing x and y
INDEX(M):	INTEGER*2 Zip code index array --

```
                    INDEX(I) is the original index of the zip code
                    with centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):             INTEGER*2
                    Pointer array --
                    PNT(I) points to the beginning of information on
                    the zip code with centroid (XCENT(I),YCENT(I)) in
                    the zip code boundary file
ZIP5(M):            INTEGER*4
                    Array of five-digit zip codes, in original
                    sequential order
ZIPCLR(M):          INTEGER*2
                    Zip code color array for POD site types,
                    if previously set
MAXPLT:             INTEGER*4
                    Maximum number of plotting points allowed in a
                    map region

Output Variables --
IERR:               INTEGER*2
                    Error flag --
                    IERR=0 for normal return
                    IERR=1 if an error was encountered
PODSET:             LOGICAL
                    Flag indicating whether or not POD colors were
                    successfully set
ZIPCLR(M):          INTEGER*2
                    Zip code color array for POD site types
XPLOT(2*MAXPLT):    REAL*4
                    Work array containing x coordinates of points to
                    plot in a map region
YPLOT(2*MAXPLT):    REAL*4
                    Work array containing y coordinates of points to
                    plot in a map region
LONG(2*MAXPLT):     REAL*4
                    Work array
PERM(2*MAXPLT):     INTEGER*2
                    Work array

Programs Called --
DISPLY
PAUSE

Calling Programs --
DRIVER
```

SUBROUTINE RANGES(RNGSET,NRANGE,RANGE,WRKMIN,WRKMAX):

This subroutine sets ranges for the workload display map and allows the user to make changes.

```
Source Code Location --
IOSUBS.FOR

Input Variables --
```

```
RNGSET:           LOGICAL
                  Flag indicating whether or not ranges have been
                  previously set
NRANGE:           INTEGER*4
                  Number of ranges for previous workload display
RANGE(0:NRANGE):  REAL*4
                  Array containing range limits for previous
                  workload display
WRKMIN:           INTEGER*4
                  Minimum workload value
WRKMAX:           INTEGER*4
                  Maximum workload value

Output Variables --
RNGSET:           LOGICAL
                  Flag indicating that ranges have been set
NRANGE:           INTEGER*4
                  Number of ranges for workload display
RANGE(0:NRANGE):  REAL*4
                  Array containing range limits --
                  The lower limit of the Ith range is RANGE(I-1)
                  and the upper limit is RANGE(I)

Programs Called --
READBF
ROUND

Calling Programs --
WRKMAP
```

SUBROUTINE READBF(IERR,MXITMS,NITEMS,ITEM):

This subroutine reads an input string from the terminal into a buffer and tries to convert it to a list of real numbers. The items in the string must be separated by spaces or commas.

```
Source Code Location --
IOSUBS.FOR

Input Variable --
MXITMS:       INTEGER*4
              Maximum allowed number of items in input string

Output Variables --
IERR:         INTEGER*4
              Error flag --
              IERR=0 for normal return
              IERR=1 if an error was encountered
NITEMS:       INTEGER*4
              Number of items in input string
ITEM(NITEMS): REAL*4
              Array containing list of real numbers in input
              string
```

Programs Called --
None

Calling Programs --
GEOFAC
OPTION
RANGES
SOLVE
WRKMAP

SUBROUTINE REPFIL(ID):

This subroutine initializes the report file REPORT.id. Either a new report file is created or information is appended to an existing report file. A report header is written giving the date and time. The report file is assigned to unit 12, which remains open after returning from the subroutine.

Source Code Location --
IOSUBS.FOR

Input Variable --
ID: CHARACTER*2
    District number

Programs Called --
CONVRT
GETDAT
GETTIM

Calling Programs --
DRIVER

SUBROUTINE REPPRB(M,ZIPCLR,ZIP5,DSTLIM,IRSWT,TXPWT,IMFFNC,BMFFNC,
NREC,DIFZIP,DIFPOD,DIFFAC,FXCOST,CSQRFT,SQFT,
MILCST,IRSFCT,TXPFCT,STFFCT,SWITCH,NCLASS,
STAFF1,STAFF2,STAFF3,PODCHG):

This subroutine writes information on the problem initialization into the report file.

Source Code Location --
IOSUBS.FOR

Input Variables --

M:	INTEGER*4
	Number of zip codes
ZIPCLR(M):	INTEGER*2
	Zip code color array for POD site types
ZIP5(M):	INTEGER*4
	Array of five-digit zip codes
DSTLIM:	REAL*4
	Maximum travel distance allowed from zip code to POD
IRSWT:	REAL*4

Weight assigned to IRS travel costs

TXPWT: REAL*4
Weight assigned to taxpayer travel costs

IMFFNC(2): INTEGER*4
Array indicating which IRS functions are included in the cost calculation for IMF data

BMFFNC(2): INTEGER*4
Array indicating which IRS functions are included in the cost calculation for BMF data

NREC: INTEGER*4
Number of non-unit travel difficulty factors

DIFZIP(NREC): INTEGER*4
Five-digit zip code array for use with travel difficulty factors

DIFPOD(NREC): INTEGER*4
POD zip code array for use with travel difficulty factors

DIFFAC(NREC): REAL*4
Array of travel difficulty factors

FXCOST(NPOD): REAL*4
Array containing fixed costs of POD sites

CSQRFT(NPOD): REAL*4
Array containing yearly rental costs per square foot for POD site office space

SQFT: REAL*4
Office space required per TDI-TDA (square feet)

MILCST: REAL*4
Mileage cost ($/mile)

IRSFCT(NCLASS): REAL*4
Array of IRS trip factors

TXPFCT(NCLASS): REAL*4
Array of taxpayer trip factors

STFFCT(NCLASS): REAL*4
Array of staffing factors

SWITCH(NCLASS): INTEGER*2
Array indicating which columns of the workload file are included in the cost calculation

NCLASS: INTEGER*4
Number of workload columns in workload file (not including display columns)

STAFF1: INTEGER*4
Total examination staff available

STAFF2: INTEGER*4
Total collection staff available

STAFF3: INTEGER*4
Total other staff available

PODCHG: LOGICAL
Flag indicating whether POD information has been changed

Programs Called --
OPTION

Calling Programs --

SUBROUTINE REPSOL(IERR,ZIP5,PODLST,M,ZIPIND,ZIPCLR,ID,
IMFEXM,BMFEXM,TDICOL,TDACOL,
IMFEX2,BMFEX2,TDICO2,TDACO2):

This subroutine is executed when the user chooses "F6 - Report zip code and staff assignments" from the top menu. It writes information on the problem solution into the report file.

Source Code Location --
IOSUBS.FOR

Input Variables --

ZIP5(M):	INTEGER*4 Array of five-digit zip codes, in original sequential order
M:	INTEGER*4 Number of zip codes
ID:	CHARACTER*2 District number

Output Variables --

IERR:	INTEGER*4 Error flag -- IERR=0 for normal return IERR=1 if an error was encountered
PODLST(NPOD):	INTEGER*2 Work array containing POD indices
ZIPIND(M):	INTEGER*2 Work array containing POD assignments
ZIPCLR(M):	INTEGER*2 Work array containing pointers to PODLST
IMFEXM(NPOD):	REAL*4 Work array containing adjusted POD workloads
BMFEXM(NPOD):	REAL*4 Work array containing adjusted POD workloads
TDICOL(NPOD):	REAL*4 Work array containing adjusted POD workloads
TDACOL(NPOD):	REAL*4 Work array containing adjusted POD workloads
IMFEX2(NPOD):	REAL*4 Work array containing unadjusted POD workloads
BMFEX2(NPOD):	REAL*4 Work array containing unadjusted POD workloads
TDICO2(NPOD):	REAL*4 Work array containing unadjusted POD workloads
TDACO2(NPOD):	REAL*4 Work array containing unadjusted POD workloads

Programs Called --
PAUSE

Calling Programs --

DRIVER

FUNCTION REVERS(X,MIN,MAX):

This function reverses the direction of a coordinate axis. (This is necessary because the x coordinates in the GDT files increase from right to left, but GKS assumes that they increase from left to right.)

Source Code Location --
MAPSUBS.FOR

Input Variables --
X: REAL*4
Value to be reversed
MIN: REAL*4
Minimum value of axis to be reversed
MAX: REAL*4
Maximum value of axis to be reversed

Output Variable --
REVERS: REAL*4
Reversed value of x

Programs Called --
None

Calling Programs --
MAPKEY
MARKZP
STEMAP
ZIPMAP
ZOOMIN

FUNCTION ROUND(X):

This subroutine rounds a real number to the nearest integer. Numbers ending in .5 are truncated.

Source Code Location --
IOSUBS.FOR

Input Variable --
X: REAL*4
Real number

Output Variable --
ROUND: INTEGER*4
Rounded value of x

Programs Called --
None

Calling Programs --
OPTION

RANGES
WRKMAP

SUBROUTINE SEARCH(XPT,YPT,M,X,Y,POS):

This subroutine searches the list of sorted centroids to find the centroid closest to a given point. A binary search is done to find the nearest x coordinate, and then a sequential search is done to find the nearest y coordinate.

Source Code Location --
MAPSUBS.FOR

Input Variables --
XPT: REAL*4
X-coordinate of point to be located
YPT: REAL*4
Y-coordinate of point to be located
M: INTEGER*4
Number of zip codes
X(M): REAL*4
Array containing x-coordinates of centroids, sorted according to increasing x and y
Y(M): REAL*4
Array containing y-coordinates of centroids, sorted according to increasing x and y

Output Variable --
POS: INTEGER*4
Position of (XPT,YPT) in centroid list
(i.e. (XPT,YPT)=(X(POS),Y(POS)))
POS=0 if point was not found

Programs Called --
None

Calling Programs --
MAPKEY

SUBROUTINE SETCOL(WKID,PALETT):

This subroutine defines the colors to be used by the display device. The type of display device in use is determined by a call to the GKS routine GQCF which returns the number of colors available on the current device driver. For a CGA display in medium-resolution four-color mode, this subroutine sets palette two with a blue background. For an EGA display, it sets sixteen colors in one of two possible palettes. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variable --
WKID: INTEGER*2

```
        Workstation identifier
PALETT: INTEGER*2
        Palette identifier (1 or 2) --
        PALETT=1: 16 distinct colors
        PALETT=2: Colors 4 through 9 are shades of green
        (Palett is not used in medium resolution mode)

Programs Called --
GQCF
GSCR

Calling Programs --
SPLTWN
WINDOW
```

SUBROUTINE SOLMAP(IERR,ID,M,XCENT,YCENT,INDEX,PNT,ZIPCLR, ZIP5,MAXPLT,XPLOT,YPLOT,LONG,PERM):

This subroutine initializes the display of a solution map. It is executed when the user chooses "F5 - Display optimal POD locations" from the top menu. It checks the solution file SOLUTION.id to see whether the graph-coloring algorithm was used by the solver. (This algorithm is used only if an input file ADJACENT.id is present.) If graph coloring was not used, then only zip codes which are POD sites (as determined by the solution) will be colored on the map (using color #3); all other zip codes will be in the background color (color #1). (Note: A negative color number indicates that the centroid will be drawn as a small box; otherwise the centroid is an asterisk.) If graph coloring was used, then a color for each zip code is read from the solution file (the color number is increased by one to avoid using the background color). The resulting map will show a POD and its assigned zip codes all in the same color. The subroutine DISPLY is called to draw the map.

```
Source Code Location --
DRIVER.FOR

Input Variables --
ID:             CHARACTER*2
                District number
M:              INTEGER*4
                Number of zip codes
XCENT(M):       REAL*4
                Array containing x-coordinates of centroids,
                sorted according to increasing x and y
YCENT(M):       REAL*4
                Array containing y-coordinates of centroids,
                sorted according to increasing x and y
INDEX(M):       INTEGER*2
                Zip code index array --
                INDEX(I) is the original index of the zip code
                with centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):         INTEGER*2
                Pointer array --
                PNT(I) points to the beginning of information on
```

```
                    the zip code with centroid (XCENT(I),YCENT(I)) in
                    the zip code boundary file
ZIP5(M):            INTEGER*4
                    Array of five-digit zip codes
MAXPLT:             INTEGER*4
                    Maximum number of plotting points allowed in a
                    map region

Output Variables --
IERR:               INTEGER*4
                    Error flag --
                    IERR=0 for normal return
                    IERR=1 if an error was encountered
ZIPCLR(M):          INTEGER*2
                    Zip code color array
XPLOT(2*MAXPLT):    REAL*4
                    Work array containing x coordinates of points to
                    plot in a map region
YPLOT(2*MAXPLT):    REAL*4
                    Work array containing y coordinates of points to
                    plot in a map region
LONG(2*MAXPLT):     REAL*4
                    Work array
PERM(2*MAXPLT):     INTEGER*2
                    Work array

Programs Called --
DISPLY
PAUSE

Calling Programs --
DRIVER
```

SUBROUTINE SOLVE(IERR,ID,PODSET,MAXPOD,PODZIP,XPOD,YPOD,
M,HOLD,FXCOST,CSQRFT,TRVDIF,ZIPCLR,ZIP5,JCOEF,
COST,MAXDIF,DIFZIP,DIFPOD,DIFFAC):

This subroutine allows the user to set parameters to initialize a location problem and then calculates costs which are passed to the solution procedure SOLVER.EXE. It is executed when the user chooses "F4 - Solve for optimal POD locations" from the top menu. First, several parameters such as cost per mile, maximum travel distance, and staff totals are read from the file PODSITE.id. Then opening/closing costs and office rental costs for each POD site are read. If any POD sites exist which are not in the file, the user is asked to enter the costs. Then trip factors and staffing factors are read from the file WORKLOAD.id. The user is allowed to set weights for IRS and taxpayer costs, to turn categories of workload on or off, to change the distance limit and staff totals, and to set travel difficulty factors. The parameter settings are summarized in the report file REPORT.id.

After all parameters have been set, the program proceeds to calculate costs of assigning zip codes to POD sites. These costs are passed to the PASCAL solution procedure in the unformatted sequential file TRANSFER.id. For each zip code, the program finds all possible POD sites (current, potential and

fixed) within the specified distance limit and calculates the cost of each assignment using the function COSTFN. The assignments are then sorted in order of decreasing cost; if a zip code is a possible POD site, its cost is last in the list. The list of costs and POD sites for a particular zip code is written as one record of the file TRANSFER.id. The type of the zip code (i.e., whether it is a potential POD site, a fixed POD site, etc.) is also included in the record. The record is padded with zeroes to make the file readable by a PASCAL program. If, for some zip code, there are no possible POD sites within the distance limit, an error message will appear on the screen. Otherwise, the subroutine is exited and control passes to the solution procedure. (The solution algorithm is described in "The Internal Revenue Service Post-of-Duty Location Modeling System: Programmer's Manual for PASCAL Solver".)

Source Code Location --
DRIVER.FOR

Input Variables --

ID:	CHARACTER*2 District number
PODSET:	LOGICAL Flag indicating whether or not POD colors have been previously set
MAXPOD:	INTEGER*4 Maximum possible number of POD sites
M:	INTEGER*4 Number of zip codes
ZIPCLR(M):	INTEGER*2 Zip code color array for POD site types, if previously set
ZIP5(M):	INTEGER*4 Array of five-digit zip codes, in original sequential order
MAXDIF:	INTEGER*4 Maximum number of non-unit travel difficulty factors

Output Variables --

IERR:	INTEGER*4 Error flag -- IERR=0 for normal return IERR=1 if an error was encountered
PODZIP(NPOD):	INTEGER*2 Array containing zip code indices of POD sites
XPOD(NPOD):	REAL*4 Array containing x-coordinates of centroids of POD sites
YPOD(NPOD):	REAL*4 Array containing y-coordinates of centroids of POD sites
HOLD(NPOD):	REAL*4 Work array
FXCOST(NPOD):	REAL*4 Array containing fixed costs of POD sites

```
CSQRFT(NPOD):    REAL*4
                 Array containing yearly rental costs per square
                 foot for POD site floor space
TRVDIF(NPOD):    REAL*4
                 Work array containing travel difficulty factors
                 from a zip code to all possible POD sites
ZIPCLR(M):       INTEGER*2
                 Zip code color array for POD site types
                 -1 <=> Standard zip code
                 -2 <=> Potential POD site
                 -3 <=> Newly added currently existing POD site
                  3 <=> Standard currently existing POD site
                 -4 <=> Fixed POD site with opening cost
                  4 <=> Fixed POD site without closing cost
JCOEF(NPOD):     INTEGER*2
                 Work array containing POD indices for a zip code
COST(NPOD):      REAL*8
                 Work array containing POD costs for a zip code
DIFZIP(NREC):    INTEGER*4
                 Five-digit zip code array for use with travel
                 difficulty factors
DIFPOD(NREC):    INTEGER*4
                 POD zip code array for use with travel difficulty
                 factors
DIFFAC(NREC):    REAL*4
                 Array of travel difficulty factors --
                 DIFFAC(I) is the factor associated with zip code
                 DIFZIP(I) and POD site DIFPOD(I)

Programs Called --
COSTFN
GCDIST
GEOFAC
OPTION
PAUSE
READBF
REPPRB
SORT

Calling Programs --
DRIVER
```

SUBROUTINE SORT(M,X,Y,INDEX,PNT):

This subroutine uses a heapsort algorithm to sort the centroids according to increasing x and y coordinates. It is also used to sort other arrays. For a description of the heapsort algorithm, see, for example, An Introduction to Data Structures with Applications by J. P. Tremblay and P. G. Sorenson (McGraw-Hill, 1976, p. 475).

```
Source Code Location --
DRIVER.FOR

Input Variables --
```

```
M:          INTEGER*4
            Number of zip codes
X(M):       REAL*4
            Array containing x-coordinates of centroids
Y(M):       REAL*4
            Array containing y-coordinates of centroids
INDEX(M):   INTEGER*2
            Zip code index array --
            INDEX(I)=I on input
PNT(M):     INTEGER*2
            Pointer array --
            PNT(I) points to the beginning of information on the
            Ith zip code in the zip code boundary file

Output Variables --
X(M):       REAL*4
            Array containing x-coordinates of centroids,
            sorted according to increasing x and y
Y(M):       REAL*4
            Array containing y-coordinates of centroids,
            sorted according to increasing x and y
INDEX(M):   INTEGER*2
            Zip code index array --
            INDEX(I) is the original index of the zip code with
            centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):     INTEGER*2
            Pointer array, sorted corresponding to XCENT and
            YCENT arrays

Programs Called --
None

Calling Programs --
CENSRT
SOLVE
THIN
```

SUBROUTINE SPLTWN(WKID,RTRN,PALETT,XMIN,XMAX,YMIN,YMAX):

This subroutine sets up a split window and viewport (designed for a menu on the left side and a map on the right side). World coordinates for the left side must be between 0 and 1 and use transformation number 7. The width of the left window is set to be the width of eight characters. World coordinates and a transformation number for the right side must be specified by the programmer. The aspect ratio of the map is adjusted for latitude-longitude data. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID:    INTEGER*2
         Workstation identifier
RTRN:    INTEGER*2
```

```
        Right transformation number (between 1 and 6)
PALETT: INTEGER*2
        Palett identifier (1 or 2)
XMIN:   REAL*4
        Smallest x value of right world coordinates
XMAX:   REAL*4
        Largest x value of right world coordinates
YMIN:   REAL*4
        Smallest y value of right world coordinates
YMAX:   REAL*4
        Largest y value of right world coordinates

Programs Called --
BORDER
GCDIST
GCLRWK
GQMDS
GQTXX
GSELNT
GSVP
GSWKVP
GSWKWN
GSWN
SETCOL
WINDOW

Calling Programs --
STEMAP
ZIPMAP
```

SUBROUTINE STEMAP(IERR,TRN,PALETT,ID,XMIN,XMAX,YMIN,YMAX,
M,XCENT,YCENT,INDEX,ZIPCLR,MAXPLT,XPLOT,YPLOT,
LONG,PERM):

This subroutine draws a state map, showing the outline of the state and centroids of zip codes. (Zip code boundaries are not drawn.) The state border is read from the file OUTLIN.id.

```
Source Code Location --
MAPSUBS.FOR

Input Variables --
TRN:            INTEGER*2
                Transformation number
PALETT:         INTEGER*2
                Palette identifier (1 or 2)
ID:             CHARACTER*2
                District number
M:              INTEGER*4
                Number of zip codes
XCENT(M):       REAL*4
                Array containing x-coordinates of centroids
YCENT(M):       REAL*4
                Array containing y-coordinates of centroids
```

```
        INDEX(M):          INTEGER*2
                           Zip code index array
        ZIPCLR(M):         INTEGER*2
                           Zip code color array --
                           Positive value: Centroid drawn as asterisk
                           Negative value: Centroid drawn as small box
        MAXPLT:            INTEGER*4
                           Maximum number of plotting points allowed in a
                           map region

        Output Variables --
        IERR:              INTEGER*4
                           Error flag --
                           IERR=0 for normal return
                           IERR=1 if an error was encountered
        XMIN:              REAL*4
                           X coordinate of lower left corner of map
        XMAX:              REAL*4
                           X coordinate of upper right corner of map
        YMIN:              REAL*4
                           Y coordinate of lower left corner of map
        YMAX:              REAL*4
                           Y coordinate of upper right corner of map
        XPLOT(2*MAXPLT):   REAL*4
                           Work array containing x coordinates of points to
                           plot in a map region
        YPLOT(2*MAXPLT):   REAL*4
                           Work array containing y coordinates of points to
                           plot in a map region
        LONG(2*MAXPLT):    REAL*4
                           Work array
        PERM(2*MAXPLT):    INTEGER*2
                           Work array

        Programs Called --
        BEEP
        CLSGKS
        GPL
        GPM
        GQCF
        GSMK
        GSPLCI
        GSPMCI
        MATCH
        REVERS
        SPLTWN
        THIN

        Calling Programs --
        DISPLY
```

SUBROUTINE STYLE(IERR,WKID,PALETT,INDEX):

This subroutine sets one of sixteen area-filling interior styles. For a CGA display in medium-resolution four-color mode, the style is either a solid color or a cross-hatch pattern. For an EGA display, the style is always a solid color. (See the GKS manual for definitions of graphics terms.)

```
Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID:   INTEGER*2
        Workstation identifier
PALETT: INTEGER*2
        Palette identifier (1 or 2) --
        PALETT=1: 16 distinct colors
        PALETT=2: 6 shades of green plus 10 other colors
        (Palett is not used in medium resolution mode)
INDEX:  INTEGER*4
        Index (between 1 and 16) of desired interior style

Output Variable --
IERR:   INTEGER*4
        Error flag --
        IERR=0 for normal return
        IERR=1 if an error was encountered

Programs Called --
CLSGKS
GQCF
GSFACI
GSFAIS
GSFASI

Calling Programs --
ERSMNU
MAPKEY
MENU20
MENU21
MENU22
MENU23
ZIPMAP
```

SUBROUTINE THIN(XPLOT,YPLOT,NPTS,MAXPLT,IPTS,WORK,PERM):

This subroutine thins zip code boundaries which have more than the maximum allowed number of plotting points.

```
Source Code Location --
MAPSUBS.FOR

Input Variables --
XPLOT(NPTS):    REAL*4
                Array containing x coordinates of boundary points
YPLOT(NPTS):    REAL*4
                Array containing y coordinates of boundary points
```

```
NPTS:           INTEGER*4
                Number of boundary points
MAXPLT:         INTEGER*4
                Maximum number of boundary points to be plotted
IPTS:           INTEGER*4
                Number of interior points

Output Variables --
XPLOT(NPTS):    REAL*4
                Array containing x coordinates of thinned boundary
                points
YPLOT(NPTS):    REAL*4
                Array containing y coordinates of thinned boundary
                points
NPTS:           INTEGER*4
                Number of thinned boundary points
WORK(2*MAXPLT): REAL*4
                Work array
PERM(2*MAXPLT): INTEGER*2
                Work array

Programs Called --
SORT

Calling Programs --
MAPKEY
STEMAP
ZIPMAP
```

SUBROUTINE TOPMNU:

This subroutine writes the text for the top-level menu on the screen. The format of the menu is slightly different depending on whether a CGA or an EGA display is being used. This is determined by calling the GKS routine GQCF which finds the number of colors available on the current device driver.

```
Source Code Location --
MENUSUBS.FOR

Programs Called --
BORDER
GPL
GQCF
GSPLCI
GSTXCI
GTX
WINDOW

Calling Programs --
DRIVER
```

SUBROUTINE WINDOW(WKID,TRN,PALETT,XMIN,XMAX,YMIN,YMAX):

This subroutine sets a window and a viewport in a way that preserves the aspect ratio of the world coordinates. (See the GKS manual for definitions of graphics terms.)

Source Code Location --
GKSUTIL.FOR

Input Variables --
WKID: INTEGER*2
Workstation identifier
TRN: INTEGER*2
Transformation number (between 1 and 7)
PALETT: INTEGER*2
Palette identifier (1 or 2)
XMIN: REAL*4
Smallest x value of world coordinates
XMAX: REAL*4
Largest x value of world coordinates
YMIN: REAL*4
Smallest y value of world coordinates
YMAX: REAL*4
Largest y value of world coordinates

Programs Called --
GQMDS
GSELNT
GSVP
GSWKVP
GSWKWN
GSWN
SETCOL

Calling Programs --
SPLTWN
TOPMNU

SUBROUTINE WRKMAP(IERR,ID,M,XCENT,YCENT,INDEX,PNT,ZIPCLR,
ZIP5,WORK,MAXPLT,XPLOT,YPLOT,LONG,PERM):

This subroutine initializes the display of a workload map. It is executed when the user chooses "F2 - Display workload" from the top menu. It first asks the user to choose the category of workload to be displayed and then sets a series of switches corresponding to the desired columns of the workload file WORKLOAD.id. These columns are read and summed for each zip code. Then the workloads are divided into six equally spaced ranges (or the user may specify other ranges). Each zip code is assigned a color according to the range its workload falls into. The colors are passed in an array to the subroutine DISPLY which draws the map.

Source Code Location --
DRIVER.FOR

Input Variables --
ID: CHARACTER*2

```
                  District number
M:                INTEGER*4
                  Number of zip codes
XCENT(M):         REAL*4
                  Array containing x-coordinates of centroids,
                  sorted according to increasing x and y
YCENT(M):         REAL*4
                  Array containing y-coordinates of centroids,
                  sorted according to increasing x and y
INDEX(M):         INTEGER*2
                  Zip code index array --
                  INDEX(I) is the original index of the zip code
                  with centroid (XCENT(I),YCENT(I)) before sorting
PNT(M):           INTEGER*2
                  Pointer array --
                  PNT(I) points to the beginning of information on
                  the zip code with centroid (XCENT(I),YCENT(I)) in
                  the zip code boundary file
ZIP5(M):          INTEGER*4
                  Array of five-digit zip codes, in original
                  sequential order
MAXPLT:           INTEGER*4
                  Maximum number of plotting points allowed in a
                  map region

Output Variables --
IERR:             INTEGER*4
                  Error flag --
                  IERR=0 for normal return
                  IERR=1 if an error was encountered
ZIPCLR(M):        INTEGER*2
                  Zip code color array for workload
WORK(M):          INTEGER*4
                  Work array containing workload for each zip code
XPLOT(2*MAXPLT): REAL*4
                  Work array containing x coordinates of points to
                  plot in a map region
YPLOT(2*MAXPLT): REAL*4
                  Work array containing y coordinates of points to
                  plot in a map region
LONG(2*MAXPLT):  REAL*4
                  Work array
PERM(2*MAXPLT):  INTEGER*2
                  Work array

Programs Called --
DISPLY
PAUSE
RANGES
READBF
ROUND

Calling Programs --
DRIVER
```

SUBROUTINE ZIPBND(IERR,ID,XMIN,YMIN,XMAX,YMAX,MAXZPS,NZIPS,
ZIP5,PNTR,MAXPTS,X,Y):

This subroutine converts the GDT zip code boundary file (in URBAN format) to binary direct access form. For zip codes which contain holes, it reorders the interior points so the connector segment will be as short as possible. It also inserts pointers in each zip code header record (Type 3 record) to point to the location of further information on the same zip code, if any.

Source Code Location --
GDTINST1.FOR

Input Variables --

ID:	CHARACTER*2
	Two-digit district identification number
XMIN:	REAL*4
	Minimum x coordinate of state map
YMIN:	REAL*4
	Minimum y coordinate of state map
XMAX:	REAL*4
	Maximum x coordinate of state map
YMAX:	REAL*4
	Maximum y coordinate of state map
MAXZPS:	INTEGER*4
	Maximum number of zip codes allowed
MAXPTS:	INTEGER*4
	Maximum number of points allowed in a map region

Output Variables --

IERR:	INTEGER*4
	Error flag --
	IERR=0 for normal return
	IERR=1 if an error was encountered
NZIPS:	INTEGER*4
	Number of distinct zip codes in boundary file
ZIP5(NZIPS):	INTEGER*4
	Array of five-digit zip code numbers, in sequential order
PNTR(NZIPS):	INTEGER*4
	Pointer into boundary file --
	PNTR(I) is the record number of the beginning of information on zip code ZIP5(I)
X(MAXPTS):	REAL*4
	Work array containing x coordinates of a map region
Y(MAXPTS):	REAL*4
	Work array containing y coordinates of a map region

Programs Called --
ORDER

Calling Programs --
GDTINST1

SUBROUTINE ZIPMAP(IERR,TRN,PALETT,ID,MENU,ZOOM,COLOR,XMIN,XMAX,

YMIN,YMAX,M,XCENT,YCENT,INDEX,PNT,ZIPCLR,
MAXPLT,XPLOT,YPLOT,LONG,PERM):

This subroutine draws a zip code map either for a full state or for a smaller area. Centroids are included, and the zip codes are colored if specified. The zip code boundaries are read from the file ZIPCODE.id. If the whole state is not to be drawn, but only a smaller region specified by a zoom rectangle, a search is done first to find the first and last zip codes (as given in the sorted centroid list) included in the rectangle.

Source Code Location --
MAPSUBS.FOR

Input Variables --

TRN:	INTEGER*2 Transformation number
PALETT:	INTEGER*2 Palette identifier (1 or 2)
ID:	CHARACTER*2 District number
MENU:	INTEGER*2 Menu type switch
ZOOM:	LOGICAL Zoom indicator -- ZOOM=.FALSE. if entire state is to be drawn ZOOM=.TRUE. if only part of state is to be drawn
COLOR:	LOGICAL Color indicator -- COLOR=.TRUE. if zip codes are to be colored as they are drawn COLOR=.FALSE. if just zip code boundaries are to be drawn
XMIN:	REAL*4 X coordinate of lower left corner of zoom rectangle (used only if ZOOM=.TRUE.)
XMAX:	REAL*4 X coordinate of upper right corner of zoom rectangle (used only if ZOOM=.TRUE.)
YMIN:	REAL*4 Y coordinate of lower left corner of zoom rectangle (used only if ZOOM=.TRUE.)
YMAX:	REAL*4 Y coordinate of upper right corner of zoom rectangle (used only if ZOOM=.TRUE.)
M:	INTEGER*4 Number of zip codes
XCENT(M):	REAL*4 Array containing x-coordinates of centroids
YCENT(M):	REAL*4 Array containing y-coordinates of centroids
INDEX(M):	INTEGER*2 Zip code index array
PNT(M):	INTEGER*2 Pointer array --

```
                  PNT(I) points to the beginning of information on
                  the zip code with centroid (XCENT(I),YCENT(I)) in
                  the zip code boundary file
ZIPCLR(M):        INTEGER*2
                  Zip code color array --
                  (Colors used only if COLOR=.TRUE.)
                  Positive value: Centroid drawn as asterisk
                  Negative value: Centroid drawn as small box
MAXPLT:           INTEGER*4
                  Maximum number of plotting points allowed in a
                  map region

Output Variables --
IERR:             INTEGER*4
                  Error flag --
                  IERR=0 for normal return
                  IERR=1 if an error was encountered
XMIN:             REAL*4
                  X coordinate of lower left corner of map
                  (same as input XMIN if ZOOM=.TRUE.)
XMAX:             REAL*4
                  X coordinate of upper right corner of map
                  (same as input XMAX if ZOOM=.TRUE.)
YMIN:             REAL*4
                  Y coordinate of lower left corner of map
                  (same as input YMIN if ZOOM=.TRUE.)
YMAX:             REAL*4
                  Y coordinate of upper right corner of map
                  (same as input YMAX if ZOOM=.TRUE.)
XPLOT(2*MAXPLT): REAL*4
                  Work array containing x coordinates of points to
                  plot in a map region
YPLOT(2*MAXPLT): REAL*4
                  Work array containing y coordinates of points to
                  plot in a map region
LONG(2*MAXPLT):  REAL*4
                  Work array
PERM(2*MAXPLT):  INTEGER*2
                  Work array

Programs Called --
BEEP
BORDER
CLSGKS
GFA
GPL
GPM
GQCF
GQTXX
GSMK
GSPLCI
GSPMCI
GSTXCI
GTX
```

REVERS
SPLTWN
STYLE
THIN

Calling Programs --
DISPLY
ZOOMIN

SUBROUTINE ZOOMIN(IERR,TRN,PALETT,ID,MENU,XMIN,XMAX,YMIN,YMAX,
M,XCENT,YCENT,INDEX,PNT,ZIPCLR,
MAXPLT,XPLOT,YPLOT,LONG,PERM):

This subroutine allows the user to draw a box on a previously drawn map and then zooms in on the boxed area.

Source Code Location --
MAPSUBS.FOR

Input Variables --

TRN:	INTEGER*2 Transformation number
PALETT:	INTEGER*2 Palette identifier (1 or 2)
ID:	CHARACTER*2 District number
MENU:	INTEGER*2 Menu type switch
XMIN:	REAL*4 X coordinate of lower left corner of current map
XMAX:	REAL*4 X coordinate of upper right corner of current map
YMIN:	REAL*4 Y coordinate of lower left corner of current map
YMAX:	REAL*4 Y coordinate of upper right corner of current map
M:	INTEGER*4 Number of zip codes
XCENT(M):	REAL*4 Array containing x-coordinates of centroids
YCENT(M):	REAL*4 Array containing y-coordinates of centroids
INDEX(M):	INTEGER*2 Zip code index array
PNT(M):	INTEGER*2 Pointer array -- PNT(I) points to the beginning of information on the zip code with centroid (XCENT(I),YCENT(I)) in the zip code boundary file
ZIPCLR(M):	INTEGER*2 Zip code color array
MAXPLT:	INTEGER*4 Maximum number of plotting points allowed in a map region

```
Output Variables --
IERR:              INTEGER*4
                   Error flag --
                   IERR=0 for normal return
                   IERR=1 if an error was encountered
                   IERR=2 if zoom was cancelled by user
TRN:               INTEGER*2
                   Transformation number
XMIN:              REAL*4
                   X coordinate of lower left corner of zoom region
XMAX:              REAL*4
                   X coordinate of upper right corner of zoom region
YMIN:              REAL*4
                   Y coordinate of lower left corner of zoom region
YMAX:              REAL*4
                   Y coordinate of upper right corner of zoom region
XPLOT(2*MAXPLT):   REAL*4
                   Work array containing x coordinates of points to
                   plot in a map region
YPLOT(2*MAXPLT):   REAL*4
                   Work array containing y coordinates of points to
                   plot in a map region
LONG(2*MAXPLT):    REAL*4
                   Work array
PERM(2*MAXPLT):    INTEGER*2
                   Work array

Programs Called --
CRSBOX
ERSMNU
GCLRWK
GINST
GQCF
GQMDS
GRQST
GSELNT
GSTXCI
GTX
REVERS
ZIPMAP

Calling Programs --
DISPLY
```

Section V: Changing the Program to Correspond to Changes in the Workload File Format

The current version of the POD location system (5.0) assumes that the workload file WORKLOAD.id contains three header records followed by a workload record for each zip code. Each workload record contains a zip code followed by 36 items: 14 BMF Examination entries, 10 IMF Examination entries, 4 BMF Collection entries, 2 IMF Collection entries, and 6 entries for display purposes only. From year to year this format may change, and if so, some subroutines in the POD location system must be revised. The following routines may need to be changed.

Subroutine COMP in file GDTINST1.FOR --

```
Line 293:       DO 200 I=1,3
Line 294:          READ(2,*)
Line 295:   200 CONTINUE
```

This loop skips over the three header records at the beginning of the workload file. It only needs to be changed if the number of header records changes.

Function COSTFN in file DRIVER.FOR --

```
Line 333:       DO 500 I=25,30
Line 334:          TDITDA=TDITDA+WRK(I)
Line 335:   500 CONTINUE
```

This loop calculates the total Collection workload for a zip code, read from columns 25 through 30 of the workload file. The loop should be revised if the Collection columns change.

Subroutine SOLVE in file DRIVER.FOR --

```
Line 799:       PARAMETER(NCLASS=30)
```

The parameter NCLASS defines the number of categories of workload in the workload file, not including display-only categories.

```
Line 977:       READ(10,*)TEMP,(IRSFCT(I),I=1,NCLASS)
Line 978:       READ(10,*)TEMP,(TXPFCT(I),I=1,NCLASS)
Line 979:       READ(10,*)TEMP,(STFFCT(I),I=1,NCLASS)
```

These lines read the three header records from the workload file and only need to be changed if the number of header records changes.

Subroutine WRKMAP in file DRIVER.FOR --

```
Line 1415:       CHARACTER TITLE*20,VECTOR*36
Line 1424:       PARAMETER(NCLASS=36)
```

The "36" in these lines represents the total number of workload categories in the workload file, including display-only categories.

```
Line 1460:       DO 200 I=1,24
Line 1461:          SWITCH(I)=1
Line 1462:   200 CONTINUE
```

This loop turns on a switch for each category of Examination workload, read from columns 1 through 24 of the workload file. The loop should be revised if the Examination columns change.

```
Line 1465:        DO 202 I=25,30
Line 1466:           SWITCH(I)=1
Line 1467:    202 CONTINUE
```

This loop turns on a switch for each category of Collection workload, read from columns 25 through 30 of the workload file. The loop should be revised if the Collection columns change.

```
Line 1469:  PRINT *,'-------------- COLUMN DESCRIPTIONS ---------------'
Line 1470:  PRINT *,' 1-14 BMF EXAMINATION      26,28 BMF TDA COLLECTION'
Line 1471:  PRINT *,'15-24 IMF EXAMINATION        29 IMF TDI COLLECTION'
Line 1472:  PRINT *,'25,27 BMF TDI COLLECTION     30 IMF TDA COLLECTION'
Line 1473:  PRINT *,'31-36 OTHER WORKLOAD FOR DISPLAY PURPOSES ONLY'
```

(Lines 1500-1504 are identical.)

These lines should be modified to correspond to any changes in the workload file format.

```
Line 1477:  PRINT *,'Enter number (from 1 to 36 of column to be ',
Line 1478: +         'displayed: '
Line 1492:  PRINT *,'Error -- please enter a number between 1 and 36.'
```

The "36" in these lines represents the total number of workload categories in the workload file, including display-only categories.

```
Line 1509:  PRINT *,'         111111111122222222223333333'
Line 1510:  PRINT *,'123456789012345678901234567890123456'
```

These lines print column labels for each category of workload, including display-only columns.

```
Line 1564:        DO 600 I=1,3
Line 1565:           READ(10,*)
Line 1566:    600 CONTINUE
```

This loop skips over the three header records at the beginning of the workload file. It only needs to be changed if the number of header records changes.

Subroutine OPTION in file IOSUBS.FOR --

```
Line 496:        DO 200 I=1,14
Line 497:           SWITCH(I)=1
Line 498:    200 CONTINUE
```

This loop turns on a switch for each category of BMF Examination workload, read from columns 1 through 14 of the workload file. The loop should be revised if the BMF Examination columns change.

```
Line 501:        DO 201 I=15,24
Line 502:           SWITCH(I)=1
Line 503:    201 CONTINUE
```

This loop turns on a switch for each category of IMF Examination workload, read from columns 15 through 24 of the workload file. The loop should be revised if the IMF Examination columns change.

```
Line 506:        DO 202 I=25,28
Line 507:           SWITCH(I)=1
Line 508:    202 CONTINUE
```
This loop turns on a switch for each category of BMF Collection workload, read from columns 25 through 28 of the workload file. The loop should be revised if the BMF Collection columns change.

```
Line 511:        DO 203 I=29,30
Line 512:           SWITCH(I)=1
Line 513:    203 CONTINUE
```
This loop turns on a switch for each category of IMF Collection workload, read from columns 29 and 30 of the workload file. The loop should be revised if the IMF Collection columns change.

Subroutine REPPRB in file IOSUBS.FOR --

Lines 1022-1107 contain labels for each category of workload, to be listed in the report file. These labels should be updated each year.

Subroutine REPSOL in file IOSUBS.FOR --

```
Line 1160:       PARAMETER(NCLASS=30)
```
The parameter NCLASS defines the number of categories of workload in the workload file, not including display-only categories.

```
Line 1329:       DO 600 I=1,2
Line 1330:          READ(10,*)
Line 1331:    600 CONTINUE
Line 1332:       READ(10,*)TEMP,(STFFAC(I),I=1,NCLASS)
```
These lines skip over the first two header records at the beginning of the workload file and then read the staffing factors from the third line. These lines only need to be changed if the organization of the header information changes.

```
Line 1347:       DO 700 I=15,24
Line 1348:          Z1=STFFAC(I)*WRK(I)
Line 1349:          IMFEXM(ZIPCLR(POD))=IMFEXM(ZIPCLR(POD))+Z1
Line 1350:          IMFEX2(ZIPCLR(POD))=IMFEX2(ZIPCLR(POD))+WRK(I)
Line 1351:          TOTEXM=TOTEXM+Z1
Line 1352:    700 CONTINUE
```
This loop calculates staffing information for IMF Examination workload, read from columns 15 through 24 of the workload file. The loop should be revised if the IMF Examination columns change.

```
Line 1354:       DO 701 I=1,14
Line 1355:          Z1=STFFAC(I)*WRK(I)
Line 1356:          BMFEXM(ZIPCLR(POD))=BMFEXM(ZIPCLR(POD))+Z1
Line 1357:          BMFEX2(ZIPCLR(POD))=BMFEX2(ZIPCLR(POD))+WRK(I)
Line 1358:          TOTEXM=TOTEXM+Z1
Line 1359:    701 CONTINUE
```
This loop calculates staffing information for BMF Examination workload, read from columns 1 through 14 of the workload file. The loop should be revised if the BMF Examination columns change.

```
Line 1361:        Z1=STFFAC(29)*WRK(29)
Line 1362:        TDICOL(ZIPCLR(POD))=TDICOL(ZIPCLR(POD))+Z1
Line 1363:        TDICO2(ZIPCLR(POD))=TDICO2(ZIPCLR(POD))+WRK(29)
```
These lines calculate staffing information for IMF TDI Collection workload, read from column 29 of the workload file. These lines should be changed if the IMF TDI Collection column changes.

```
Line 1366:        Z1=STFFAC(25)*WRK(25)+STFFAC(27)*WRK(27)
Line 1367:        TDICOL(ZIPCLR(POD))=TDICOL(ZIPCLR(POD))+Z1
Line 1368:        TDICO2(ZIPCLR(POD))=TDICO2(ZIPCLR(POD))+WRK(25)+WRK(27)
```
These lines calculate staffing information for BMF TDI Collection workload, read from columns 25 and 27 of the workload file. These lines should be changed if the BMF TDI Collection columns change.

```
Line 1371:        Z1=STFFAC(30)*WRK(30)
Line 1372:        TDACOL(ZIPCLR(POD))=TDACOL(ZIPCLR(POD))+Z1
Line 1373:        TDACO2(ZIPCLR(POD))=TDACO2(ZIPCLR(POD))+WRK(30)
```
These lines calculate staffing information for IMF TDA Collection workload, read from column 30 of the workload file. These lines should be changed if the IMF TDA Collection column changes.

```
Line 1376:        Z1=STFFAC(26)*WRK(26)+STFFAC(28)*WRK(28)
Line 1377:        TDACOL(ZIPCLR(POD))=TDACOL(ZIPCLR(POD))+Z1
Line 1378:        TDACO2(ZIPCLR(POD))=TDACO2(ZIPCLR(POD))+WRK(26)+WRK(28)
```
These lines calculate staffing information for BMF TDA Collection workload, read from columns 26 and 28 of the workload file. These lines should be changed if the BMF TDA Collection columns change.

Subroutine ZIPMAP in file MAPSUBS.FOR --

```
Line 1090:        CHARACTER TITLE*20,VECTOR*36
```
The "36" in this line represents the total number of workload categories in the workload file, including display-only categories.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** (See instructions)	1. PUBLICATION OR REPORT NO. NISTIR 86-3473-1	2. Performing Organ. Report No.	3. Publication Date FEBRUARY 1989

4. TITLE AND SUBTITLE

The Internal Revenue Service Post-of-Duty Location Modeling System (Version 5.0) - Programmer's Manual for FORTRAN Driver

5. AUTHOR(S)

Paul D. Domich, Richard H. F. Jackson, Marjorie A. McClain

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

The Research Division
U.S. Internal Revenue Service
1201 E Street, NW
Washington, DC 20224

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program, SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

This report is a programmer's manual for a microcomputer package which was designed by the National Institute of Standards and Technology to assist the Internal Revenue Service in choosing locations for its posts-of-duty which will minimize costs to the IRS and to the taxpayer. The package was written in two sections of code, one in FORTRAN and the other in PASCAL. This manual describes the FORTRAN driver which handles graphics displays and controls input and output for the solution procedure.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

facility location, Graphical Kernel System (GKS), interactive graphics, microcomputer, personal computer

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

66

15. Price

$14.95